M8-L1 Problem 1

In this problem you will solve for $\frac{\partial L}{\partial W_2}$ and $\frac{\partial L}{\partial W_1}$ for a neural network with two input features, a hidden layer with 3 nodes, and a single output. You will use the sigmoid activation function on the hidden layer. You are provided an input sample x_0, the current weights W_1 and W_2, and the ground truth value for the sample, $t = -2$

In [1]:
```
import numpy as np

x0 = np.array([[-2], [-6]])

W1 = np.array([[-2, 1],[3, 8],[-12, 7]])
W2 = np.array([[-11, 2, 5]])

t = np.array([[-2]])
```

Define activation function and its derivative

First define functions for the sigmoid activation functions, as well as its derivative:

In [2]:
```
# YOUR CODE GOES HERE
def sigmoid(a):
    return (1/(1+ (np.exp(-a))))

def sigmoid_derivative(a):
    return ((np.exp(-a))/((1 + (np.exp(-a)))**2))
```

Forward propagation

Using your activation function, compute the output of the network y using the sample x_0 and the provided weights W_1 and W_2

In [3]:
```
# YOUR CODE GOES HERE
a_1 = np.dot(x0.T,W1.T)
a_2 = sigmoid(a_1)
a_3 = np.dot(a_2,W2.T)
y = sigmoid(a_3)
print(f'Output for forward propogation is y = {y}')
```

```
Output for forward propogation is y = [[0.21228075]]
```

Backpropagation

Using your calculated value of y, the provided value of t, your σ and σ' function, and the provided weights W_1 and W_2, compute the gradients $\frac{\partial L}

{\partial W_2}$ and $\frac{\partial L}{\partial W_1}$.

In [4]:
```
# YOUR CODE GOES HERE

delta_2 = -1*(t-y)*sigmoid_derivative(a_3)
dL_dW2 = delta_2*a_2.T
print(f'Gradient for W2 is: \n {dL_dW2.T}')

delta_1 = delta_2*W2*sigmoid_derivative(a_1)
dL_dW1 = delta_1*x0
print(f'Gradient for W1 is: \n {dL_dW1.T} \n')
```

```
Gradient for W2 is:
 [[4.40970168e-02 1.30683358e-24 5.63406208e-09]]
Gradient for W1 is:
 [[ 8.54491518e-01  2.56347456e+00]
 [-5.22733433e-24 -1.56820030e-23]
 [-5.63406199e-08 -1.69021860e-07]]
```

In [ ]: